📚 The Prompt Engineer's Compendium

Complete Guide to Expert Prompting Across All Major AI Platforms

Section I — Foundations of Expert Prompting

Unifying Principles

1. Clarity, Specificity, Context

Core Truth: Vague prompts ⇒ generic outputs

Always add: audience, length, format, tone, and constraints to your prompts.

```
❌ Bad: "Find a restaurant"
✅ Good: "Best Indian restaurant in Cambridge, Massachusetts,
within walking distance of Harvard Yard."
```

2. Affirmative Directives (Tell the model what TO do)

Prefer positive instructions over negatives for better comprehension.

```
✅ Better: "Diagnose the issue while refraining from any PII-
related questions."
❌ Avoid: "Do not ask for username or password."
```

3. Deliberate Structure

Use clear delimiters and sections so the model can parse reliably.

Common patterns: ### Headings, fenced blocks (""""), bullet lists, and tagged segments.

Platform Nuances:
• **Claude:** Favors XML-like tags for high-fidelity parsing (e.g., <instructions>... </instructions>, <context>...</context>, <examples>...</examples>)
• **Grok:** Responds well to Markdown structure and concise sections
• **Tip:** For complex tasks, separate rules, context, examples, and output format with clear labels

Section II — Prompting Paradigms

1 Chain-of-Thought (CoT)

Guides the model to show intermediate reasoning before the final answer.

Trigger phrases: "Let's think step by step.", "First, let's reason this out.", "Work it out step by step to ensure the right answer."

Classic demonstration (the "Apple Problem"): A direct ask often fails; appending CoT prompts the model to enumerate steps (e.g., 10 - 2 - 2 + 5 - 1) and reach the correct result.

Use for: Math, logic, planning, and multi-constraint synthesis.

```
### CoT Template
### Task
[SPECIFIC QUESTION]
```

```
### Method
Let's think step by step. Show intermediate
calculations/reasons before the final answer.

### Answer
```

2 Shot-Based Instruction (In-Context Learning)

Zero-Shot: Just the instruction. Good for simple, common tasks.

One-Shot: Add one example to set tone/format for slightly nuanced tasks.

Few-Shot (2+ examples): Best for accuracy and consistency; teaches pattern/format.

```
### Few-Shot Skeleton
### Instructions
[WHAT TO DO]

### Examples
Input: ...
Output: ...

Input: ...
Output: ...

### Now Do This
Input: [YOUR INPUT]
Output:
```

3 Persona & Role-Playing (Use with care)

Assign a role to influence tone/focus (e.g., "You are an expert sentiment classifier.").

Caveat: Misaligned personas can degrade factual or reasoning performance (e.g., irrelevant role primes the wrong context).

Mitigation — "Jekyll & Hyde" ensemble: Generate two answers (persona vs neutral) and select the better one via an evaluator prompt.

```
### Persona + Neutral Ensemble Pattern
### Version A (Persona)
<role>Senior [DOMAIN] Expert</role>
[instructions]

### Version B (Neutral)
[instructions, no role]

### Judge
You are the evaluator. Compare Answer A (persona) vs Answer B
(neutral) for correctness, completeness, and adherence to
constraints.
Return ONLY:
{
  "winner": "A|B",
  "reason": "<1-2 sentences>"
}
```

4 Advanced Reasoning Patterns

Use these when you need super-reliable reasoning, planning, or tool use.

Platforms: ChatGPT, Claude, Grok, Perplexity (research), Gemini (planning), NIM (as endpoint in ReAct).

Advanced Reasoning Pattern Templates

A) ReAct (Reason + Act)

```
You are a planner-agent that alternates THOUGHT and ACTION to solve the task. Tools available: [search()], [calculator()], [code_runner()], [db.query()]. Task: [clear goal] Rules: - Always write a THOUGHT before any ACTION. - After each ACTION, record OBSERVATION. - Stop when you have enough evidence; return FINAL ANSWER with citations/derivations. THOUGHT: ... ACTION: search("...") OBSERVATION: ... THOUGHT: ... ACTION: calculator("...") OBSERVATION: ... FINAL ANSWER: ...
```

B) Tree-of-Thought (ToT)

```
Goal: [problem]. Generate 3 distinct solution paths (Thought A/B/C). For each path: list steps, assumptions, risks. Score each on (correctness, cost, coverage) 1–5. Pick the highest-scoring path and execute it step-by-step before concluding. Return: (1) short rationale of why the chosen path wins, (2) final answer.
```

C) Self-Ask (Decompose → Solve → Compose)

```
Decompose the query into sub-questions (SQ1..SQn). Answer each succinctly. Then synthesize a final answer that cites which sub-answers support each claim. Query: [your question] Output sections: Sub-Questions • Answers • Synthesis.
```

D) Graph-of-Thoughts (GoT)

```
Transform the task into a graph of micro-tasks. Nodes: [N1..Nn] with purpose, inputs, outputs. Edges: dependencies. Execute nodes in topological order; allow back-edges for refinement. Return: the graph (adjacency list) + final result.
```

E) Self-Consistency Voting

```
Produce 5 independent reasoning paths (no reuse). Return just the final answers for each, then select the winner via a brief majority vote + 1-2 line justification.
```

F) Chain-of-Verification (CoVe)

```
Task: [claim-heavy task]. 1) Draft: Produce a concise initial answer. 2) Verify-Plan: List 5-8 targeted verification questions. 3) Verify-Answer: Answer each verification question independently. 4) Revise: Update the draft, explicitly noting changes. 5) Confidence: 0-1 with a one-line caveat. Output sections: Draft • Verification Qs • Answers • Revised Answer • Confidence.
```

G) Reflexion (Critique → Improve Loop)

```
Round 1 — Answerer: Produce your best answer to [task]. Round 2 — Critic: List concrete flaws, missing cases, weak assumptions, and style issues. Round 3 — Answerer: Revise to address every critique point. Mark
```

```
changes with ▶. Stop after one loop unless quality < A-; if so, run one
more loop. Return: Final answer + bullet list of fixes applied.
```

General Task Card Template

```
### Role <role>[Optional persona aligned to task]</role> ### Goal [One
sentence objective] ### Context [Audience, tone, length, format,
constraints] ### Examples (Optional Few-Shot) Input: ... Output: ... ###
Output Format - Return as: [specify format]
```

Section III — Platform Strategy (Strengths & Styles)

Platform	Key Strengths	Unique Use Cases	Prompting Style Highlights
ChatGPT	Versatile, general-purpose; strong for text/code and multi-task workflows	Creative writing, prototyping, structured extraction	Clear sections, explicit formats, examples/delimiters
Gemini	Multimodal + real-time; deep Google ecosystem	Photo-to-figurine (Nano Banana), image/video tasks	Structured, multi-component prompts combining text + images
Claude	Strong reasoning; massive context; safety	Long-form analysis, legal/academic, structured data	XML-tagged segments for instructions/context/examples
Grok	Fast, conversational; access to X (formerly Twitter) data	Real-time threads/posts;	Concise Markdown, iterative refinement, natural language

Platform	Key Strengths	Unique Use Cases	Prompting Style Highlights
		motion-oriented content	
Perplexity	Research-centric with citations	Up-to-date synthesis, comparative reviews	Pose as research queries with tight scope/keywords
NVIDIA NIM	Optimized inference microservices (API-first)	3D asset generation pipelines; enterprise apps	API-driven prompts inside structured requests

Match Tool to Task

- General creativity: ChatGPT or Gemini
- Long-form analysis: Claude
- Live research: Perplexity
- Fast conversational: Grok
- 3D assets & enterprise: NVIDIA NIM

Design Multi-Stage Systems

Consider persona+neutral ensembles ("Jekyll & Hyde") with an evaluator step for balanced outputs

Structure + Examples

Delimiters and few-shot patterns reduce ambiguity and improve fidelity across all platforms

Section V — Copy-Paste Prompt Library (By Task)

A) Text-to-Website & Code Generation

> **End-to-end scaffold + code**

Purpose: Complete website architecture and implementation

```
Develop an architecture and code for a [e-commerce] website with
JavaScript. Specify pages, components, state, and data flow;
generate modular code.
```

ChatGPT Gemini Claude Grok

Implement specific web feature

Purpose: Create sticky header with best practices

```
Act like a programming languages expert. I need a sticky header.
Provide a CSS + JS example and explain pitfalls (z-index, scroll
throttling).
```

ChatGPT Gemini Grok Claude

Debug and harden code

Purpose: Find bugs and provide fixes

```
Find the bug in this code: [paste code]
Explain cause and provide a minimal fix + test.
```

ChatGPT Claude Grok Gemini

Output Format Hint

```
### Constraints
- Use semantic HTML, ARIA labels, and responsive design.
- Return code blocks per file: /index.html, /styles.css, /app.js.
- Include a 60s Lighthouse checklist.
```

B) Social Media Content & Marketing

High-engagement X thread

Purpose: Create viral Twitter/X thread

```
Write a 7-tweet thread on [topic] that builds curiosity, delivers
real value, and ends with a CTA to follow. Include hooks, numbered
tips, and a summarizing last tweet.
```

Grok ChatGPT Gemini

LinkedIn post from voice notes

Purpose: Transform raw notes into professional post

```
Turn this voice-note transcript into a high-authority LinkedIn post—
punchy, no fluff, leave space for engagement. [paste text]
```

ChatGPT Gemini Claude

Multi-channel content plan

Purpose: 3-month marketing strategy

```
I'm launching a new [product]. Create a 3-month content marketing
plan across blog, social, and email. Include cadence, topics, and
CTAs.
```

ChatGPT Gemini

SEM planning

Purpose: Keyword research and ad strategy

```
Act as a digital marketing manager for [company]. Seed keywords:
[list]. Expand to keyword clusters (intent, volume guess), ad copy
angles, and negative keywords.
```

ChatGPT Gemini Perplexity

C) Presentations & Pitch Decks

Structured pitch deck plan

Purpose: Complete investor presentation outline

```
Create a presentation outline and draft content for a [10-minute
pitch deck] about [product] for an [investor] audience. Main
message: XX. Key claims: 1-3. Include slide titles, bullets, and
suggested visuals.
```

ChatGPT Claude

Slide-level refinement

Purpose: Polish individual slides

```
From the outline above, write 2-5 sentences that support this
slide's main message. Suggest visuals and a transition to the next
slide.
```

ChatGPT Claude Gemini

Slide Output Format

```
Return as:
Slide #: Title
• Key point 1
• Key point 2
Visual suggestion: ...
Transition: ...
```

D) Visual & Multimedia Creation

D1) Photo → Figurine (Nano Banana)

Purpose: Transform photo into commercial figurine render

```
Create a 1/7 scale commercialized figurine of the characters in the
picture, in a realistic style, in a real environment.
The figurine is placed on a computer desk.
```

```
The figurine has a round transparent acrylic base, with no text on
the base.
The content on the computer screen is a 3D modeling process of this
figurine.
Include a toy packaging box in the background that matches the
figurine.
Keep lighting natural and reflections realistic.
```

Gemini

D2) Short 3D Animated Scene

Purpose: Create animated video clip

```
[Style]: a short 3D animated scene in a joyful cartoon style.
[Subject]: A cute creature with snow-leopard-like fur.
[Action]: happily prances through a whimsical winter forest.
[Context]: rounded, snow-covered trees; gentle falling snowflakes;
soft ambient light.
[Extras]: simple SFX for footsteps; end on a playful leap freeze-
frame.
Duration: ~8–12 seconds.
```

Gemini (Veo) Grok

E) Product Deep Research

Comparative Research

Purpose: In-depth product comparison

```
Pro Search: Compare [Product A] vs [Product B] on features,
pros/cons, price.
Scope: focus on [industry/region]. Return a table and 5+ cited
sources. Include a 2-sentence bottom line.

Follow-ups:
- Narrow scope to sources after: 2024-01-01
- Include only primary sources and whitepapers; exclude affiliate
blogs
- What are the strongest counterarguments? Cite at least 3
- Summarize consensus vs open disagreements in 5 bullets
```

Perplexity

Research Brief

Purpose: Structured research summary

```
Labs: Build a research brief on [topic]. Sections: Background, Key
Findings, Risks/Open Questions, Further Reading (linked). Limit to
400 words; cite sources inline.
```

Perplexity

Long-Context Summarization

Purpose: Hierarchical summary with citations

```
<instructions>
  Create a hierarchical summary with section headings and bullet
points.
  Append an APA-style mini-bibliography using the provided refs.
</instructions>
<sources>
  [DOC 1]
  [DOC 2]
</sources>
<constraints>
  400–600 words; keep quotations under 20 words each.
</constraints>
```

Claude

F-J) Creative & Advanced Prompts

F) Animated Video + SFX

Purpose: Motion graphics with sound

```
Create a 10-second animated clip: a paper airplane weaving through
neon city canyons at night.
Add gentle whoosh SFX synced to turns; end with a soft chime.
Style: playful, minimal; include a witty caption and 3 on-trend
```

```
hashtags.
Deliverables: MP4 + caption text.
```

Grok

G) Worldbuilding Bible

Purpose: Complete fictional world design

```
Create a worldbuilding bible for a [genre] mini-series.
Sections: Premise (2 lines), World Rules (7 bullets), Factions (3),
Key Locales (5), Core Cast (6 with desires & secrets), Season Arc (8
beats), Visual Lookbook (10 image prompts), Soundtrack (moods +
instruments), Iconic Props (7).
Style: concise, high-signal, production-ready.
```

ChatGPT Claude

H) Brand Voice Cloner

Purpose: Extract and replicate writing style

```
Given 3 writing samples (below), infer a Style Guide with: Tone
sliders, Diction, Syntax patterns, Tropes to use/avoid, CTA
patterns, and Do/Don't examples.
Then rewrite this draft in the inferred voice with a 1-paragraph
rationale.
[SAMPLES]
[DRAFT]
```

ChatGPT Claude

I) Idea Oracle

Purpose: Generate ideas through multiple lenses

```
Topic: [space]. Generate 12 ideas via lenses: (1) First principles,
(2) Inversion, (3) Constraints-as-features, (4) Analogy, (5) Edge
cases, (6) Opposite day.
Return a 2-column table: Idea • Why it could work.
```

ChatGPT Claude Gemini

J) Cinematic Shot-List Factory

Purpose: Film production planning

```
Turn this scene idea into a 30-shot list with shot type, lens,
movement, and motivation. Tag B-roll and cutaways. Add 3 alt shots
for safety.
Scene: [description]
```

ChatGPT Gemini

Additional Creative Prompts

Fusion Style Writer

Purpose: Blend writing styles

```
Write [format] that fuses [Author/Style A] + [Author/Style B] while
avoiding direct mimicry; describe the blend in 2 lines first.
Topic: [topic]. Length: [x].
```

ChatGPT Claude

"Director's Prompt" Formula (Image)

Purpose: Detailed image generation

```
[Subject] • [Action] • [Environment] • [Composition] • [Lighting] •
[Camera/lens] • [Color/mood] • [Materials/textures] • [Post-
processing].

Example: A tiny clockwork dragon • landing on a tea cup • on a
sunlit windowsill • rule-of-thirds close-up • rim-lit with soft
bloom • 85mm shallow DoF • warm amber • brass + porcelain • subtle
film grain.
```

Gemini ChatGPT

Nano-Banana Variants

Purpose: Different figurine styles

```
Make a 1/7-scale commercial figurine of the subject from the photo
in [style]: (diorama / papercraft / LEGO brick-built / claymation).
On a tidy desk; clear acrylic round base; no text; natural
reflections.
```

Gemini

Veo Short — Action + SFX

Purpose: Short video with sound effects

```
Create an 8–12s clip: [subject] performs [action] in [setting].
Include [SFX beats] synced to motion; end on a logo sting. Output
MP4 + caption.
```

Gemini (Veo)

Hook Factory (20 Frameworks)

Purpose: Generate multiple hook variations

```
Generate 20 hooks for [topic] using frameworks: AIDA, PAS, ABT, 4U,
Before-After-Bridge, Problem-Myth-Truth, Data-Gap-Payoff, Numbered-
List, Hot-Take, Contrarian, Mini-Story, Question, Challenge, Stat-
Shock, Template-Madlib, Insider-Tip, Mistake-to-Avoid, Debunk, Case-
Study, Objection-Smash. Return a table: Hook • Framework • Angle.
```

ChatGPT Claude

Research Thread Distiller

Purpose: Convert research into social media thread

```
Turn these sources into a 7-post X thread with a bold claim, 5
supported points (1 link each), and a takeaway + CTA. Include an
alt-text line per image. [links]
```

Grok ChatGPT

Meme Engine

Purpose: Generate meme concepts

```
Propose 5 meme concepts for [topic]. For each: template name,
caption text, alt text for accessibility, and why it will resonate
with [audience].
```

ChatGPT Grok

Technical & Code Analysis Prompts

Property-Based Test Generator

Purpose: Generate comprehensive test cases

```
Given code [paste], infer properties/invariants and generate
property-based tests (e.g., Hypothesis/QuickCheck). Include edge
cases and shrinking hints.
```

ChatGPT Claude

Targeted Verification Qs for Code

Purpose: Pre-merge code review

```
For this diff, generate verification questions that would uncover
hidden bugs (nullability, bounds, concurrency, perf). Map each Q to
the exact line(s) touched. [paste diff]
```

Claude ChatGPT

Repo Mental Model

Purpose: Understand codebase architecture

```
Create a bird's-eye map of this repo: modules, data flow, hotspots,
and dependency risks. Return Mermaid diagrams + a 90-day refactor
plan. [link/tree]
```

Claude ChatGPT

NVIDIA NIM Prompt Catalog

Copy-paste prompts for build.nvidia.com "Try It!" boxes. Replace [brackets] with your values.

LLMs (Chat Completions)

1) RAG summarizer — NIM: meta/llama-3.1-70b-instruct

```
System: Summarize with explicit citations like [S1], [S2]... User: Question: [Q]. Sources: [S1 text] [S2 text] [S3 text]. Write 7 bullets; each ends with citations. Finish with a one-line verdict.
```

2) OCR → tidy JSON tables — NIM: meta/llama-3.2-11b-vision-instruct

```
"Read every visible table and return only JSON: {tables:[{title,rows:[{cells:[...]}]}]}. Preserve numbers/units; don't guess."
```

3) Chart interpreter — NIM: meta/llama-3.2-11b-vision-instruct

```
"Explain variables, trends, anomalies; give 3 causal hypotheses. End with: Caution: [bias/limitation]."
```

4) Zero-shot class label — NIM: nvidia/nvclip

```
"a red vintage roadster in studio lighting; front-three-quarter view"
```

5) Fine-grained image query — NIM: nvidia/nv-dinov2

```
"close-up of [object] featuring [detail], background [style], captured at [time of day]"
```

6) ASR Word boosting — NIM: nvidia/parakeet-ctc-1.1b-asr

```
"Boost list: AntiBERTa(40), ABlooper(40), your jargon, product. Punctuation on; profanity filter on; numbers as digits."
```

7) ASR Fast English — NIM: nvidia/parakeet-tdt-0.6b-v2

```
"Transcribe with automatic punctuation + word timestamps. Prefer standard casing."
```

8) Neutral explainer (SSML) — NIM: nvidia/magpie-tts-multilingual

```
<speak> <prosody rate="92%" pitch="+1">Welcome to <sub alias="en vee dee uh">NVIDIA</sub> NIM.</prosody> <break time="300ms"/> Today we'll cover <emphasis level="moderate">three steps</emphasis>: setup, auth, and integration. </speak>
```

9) 5-second voice-match — NIM: nvidia/magpie-tts-flow

```
"Clone the attached 5-sec reference and read this: '[script]'. Target: neutral corporate, 48kHz WAV, mono. Keep sibilance soft."
```

10) Lipsync from audio — NIM: nvidia/audio2face-3d

```
"Generate facial blendshapes from this narration. Output: ARKit-compatible curves; neutralize head motion; smooth jaw at 6 Hz."
```

3D Asset Generation Pipeline

A) Asset spec generator — NIM: nvidia/llama-3.1-nemotron-70b-instruct

```
System: You design production-ready asset specs. Output only the JSON schema shown; no prose. User: "Create specs for [theme]. For each asset, include: name, category (prop|environment|character), description, dimensions_cm, poly_budget (low|mid|high), lods (e.g., LOD0, LOD1, LOD2), materials (PBR: albedo, normal, roughness, metallic), file_format (gltf|usdz|usd), pivot (base|center), scale_units (cm), licensing_note, ref_images (keywords). Return JSON: { "assets":[ {...} ], "theme":"[theme]" }."
```

B) 3D generation — NIM: microsoft/trellis

```
"Generate a physically coherent [category] named [name] matching this description: [description]. Constraints: dimensions ≈ [dimensions_cm] cm, [poly_budget] polycount, provide [lods], PBR textures (albedo/normal/roughness/metallic), export as [file_format] with real-world cm scale, pivot at [pivot]. Ensure watertight meshes; no non-
```

```
manifold edges; correct face normals; clean UVs; texture resolution ≥ 2K."
```

C) Audio generation — NIM: nvidia/magpie-tts-zeroshot

```
"Clone the attached 5-sec sample and generate [N] lines for the asset pack '[theme]'. Tone [tone], language [lang], sample rate 48 kHz, mono WAV, loudness -16 LUFS. Lines: [bulleted lines]. Return a file list with durations."
```

D) Validation — NIM: nvidia/usdvalidate

```
"Validate these assets for OpenUSD. Enforce: cm units; pivot at [pivot]; outward normals; no non-manifold geometry; no inverted UVs; texture existence and ≥ 2048px; LOD naming LOD0/LOD1/LOD2; axis-aligned bounds; material-slot count matches spec. Return JSON report: {file, errors[], warnings[], metrics:{tris, materials, texResMin}}."
```

E) If-fail auto-regen — NIM: nvidia/llama-3.1-nemotron-70b-instruct

```
"Given this validator report [paste JSON], propose minimal constraint tweaks per asset (e.g., reduce poly, fix UV seams, raise texture res). Output JSON patch: {name, changes[], rationale}. Keep changes few and surgical."
```

F) Manifest + QA roll-up — NIM: nvidia/llama-3.1-nemotron-70b-instruct

```
"Combine the final asset specs and validator reports into a single manifest: {assets:[{name, file, dims_cm, tris, lods, materials, texResMin, issues_fixed[]}], theme, created_at}. Produce a 5-bullet QA summary."
```

🎯 Key Takeaways & Best Practices

🔄 Iteration is Key

The first prompt is